Project Report Format

INTRODUCTION

1. Project Overview
2. Purpose

LITERATURE SURVEY

3. Existing problem
4. References
5. Problem Statement Definition

IDEATION & PROPOSED SOLUTION

6. Empathy Map Canvas
7. Ideation & Brainstorming
8. Proposed Solution
9. Problem Solution fit

REQUIREMENT ANALYSIS

10. Functional requirement
11. Non-Functional requirements

PROJECT DESIGN

12. Data Flow Diagrams
13. Solution & Technical Architecture
14. User Stories

PROJECT PLANNING & SCHEDULING

15. Sprint Planning & Estimation
16. Sprint Delivery Schedule
17. Reports from JIRA

CODING & SOLUTIONING (Explain the features added in the project along with code)

18. Feature 1
19. Feature 2
20. Database Schema (if Applicable)

TESTING

21. Test Cases
22. User Acceptance Testing

RESULTS

SSN COLLEGE OF ENGINEERING, CHENNAI

ADITHYA SRIRAM R, ADITHYA R

,MEGANATHAN V, LOCHAN N

INTRODUCTION

1. Project Overview:

Personal Expense Tracker Application -

Personal Finance entails all the financial decisions and activities that a Finance App makes your life easier by helping you to manage your finances efficiently.

2. Purpose:

A personal finance app will help you with budgeting and accounting but also give you helpful insights about money management. Personal finance

applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

LITERATURE SURVEY

3. Existing problem

A personal expense tracker app is designed to help you optimize the spending and savings choices you make each month. By putting all your financial commitments and goals in one place, a budgeting app can give you better visibility into your financial choices and habits.
Similar to the apps you may use from your bank or credit union, a budgeting app may provide additional functionality such as financial goal setting and cash flow tracking across multiple financial accounts. Budgeting apps can sync with your bank and credit card accounts to provide a holistic view of your finances.

4. References

Budgeting is a process that begins with identifying your financial goals, along with observing your spending and savings habits. When you truly understand how much money is going out and coming in, you're better prepared to handle both the expected and unexpected financial challenges life brings.

The mindset you bring to managing your money is important. Establishing a budget is a critical first step to gaining control over your finances. If you've never

worked with a personal budget, it may take a few cycles for your habits to catch up. And, if you have money habits you need to improve, the right app can help.

Exploring Different Expense Tracker Apps and their Pros and Cons

Prism shows all your bills and financial accounts in a single app, giving you a complete picture of your finances. The app touts more than 11,000 billers, including larger banks and even smaller utility companies. Add your bills to the app, and Prism automatically tracks your bills and sends due date reminders to help you prevent late payments.

You can use the app to pay your bills by scheduling payments made the same day or several days in advance. Prism eliminates the need to log into multiple accounts to pay bills.

Pros

- Handy payment due date reminders
- Full picture of your accounts in one app

Cons

- Limited features - bill pay only

Many other personal finance apps are for individual use. But Spendee allows you to create shared wallets with friends and family that you can use to manage shared expenses for a household budget.

You can import your bank transactions and let the app categorize them to tally how you're spending money each month. You can also manually add cash expenses for a more accurate picture of where your money goes. And if you're concerned about going over budget, you can set budgeted amounts for each spending category and track your progress toward the budgeted amount.

Additionally, Spendee's bill tracker functionality ensures you remember to pay each of your bills and avoid late payment penalties. If you're going on a trip or another special event, you can create a category

specifically for that event to track your spending and keep yourself on budget.

Pros

- Handy spending categorisation
- Easily accessible by family members or roommates

Cons

- Free plan has limited features
- Bank account sync only available with Premium plan

Mint, Intuit’s personal finance app, is a popular app that provides your complete financial picture in one place. Once you link your credit and debit cards to your account, Mint pulls your transactions, categorizes them, and shows how you spend your money. You can keep track of your bills and spending and create a

budget you can stick to.

The site provides access to your credit score for free, and you can get a breakdown of the factors contributing to your score to stay on top of your credit health. Plus, you can track your investments and manage utility payments.

Pros

- Mint is free
- Simple bill tracking
- Handy spending categorization

Cons

- Users may find the ads annoying

Mobills organizes your expenses in categories so you can track how your spending is progressing toward your budgeted amount. See the amount you have remaining to spend in each budget category so you can rein in your spending as needed.

Mobills' budget planning app includes interactive charts that allow you to analyze your financial life; you can use them to make adjustments as you need to reach your larger financial goals. Add your credit cards to the app so you can see your current balance and spending limits all in one place. You can also add all your bills and due dates to keep track of when your bills need to be paid.

Pros

- Handy spending categorization
- Helpful visuals

Cons

- Free version has limited features

5. Problem Statement Definition

- The user will have to monitor various forms of spending as well as the withdrawal of funds from their accounts.
- Additionally, the user would require help allocating cash effectively and opening up opportunities for future investments and saves.
- When there is excessive expenditure, the email subscription will serve as a reminder, encouraging the user to make wise financial decisions.
- In order to help the customers budget more effectively in the future, it would also be necessary to provide them with a thorough analysis that highlights their spending patterns.

After understanding the functionalities of all the above personal finance apps, we can safely come up with a list of points that are important to make a good personal finance app.

- Ease of use.
- A quick overview of all user finances.
- Beautiful UI/UX.
- Graphical representation of monthly spend
- Actionable saving tips and financial advice.
- Being able to set a threshold limit for monthly expenditure
- Automatic warnings when the user goes past the threshold limit
- Live updates on any financial activity.

Common issues that we came across in the above apps were:-

- Too Expensive
- Too many advertisements
- Not visual enough

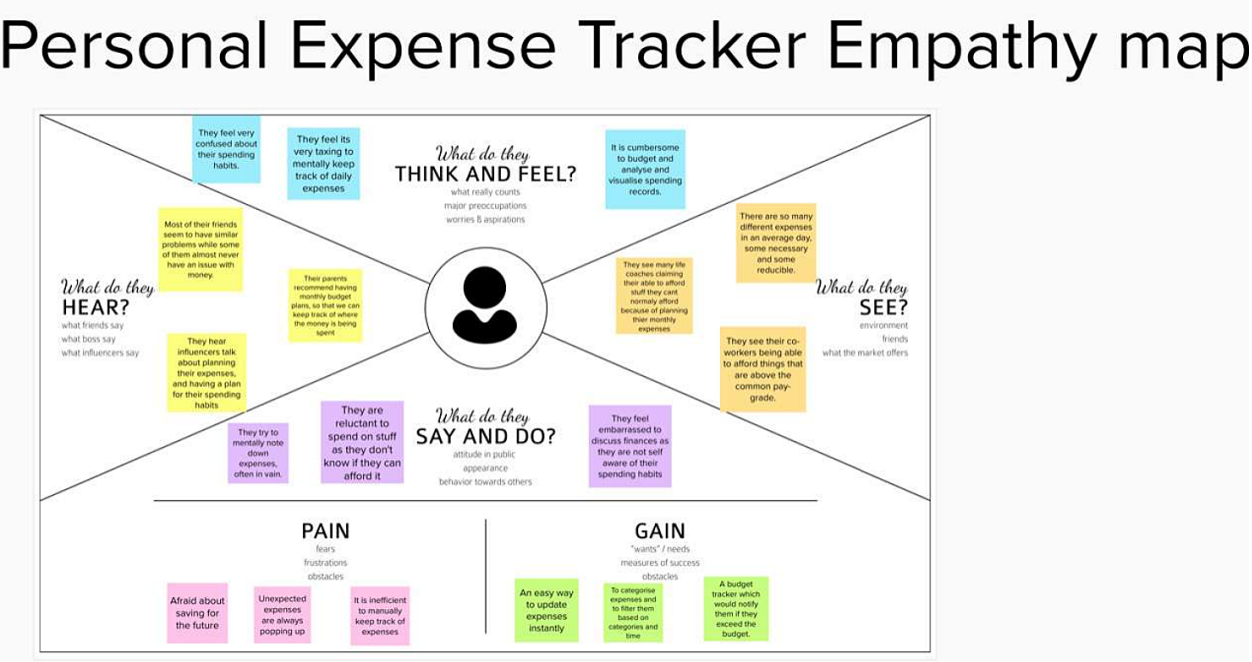
- Too many features that overwhelm the user
- Free plans have only few features

References(links)

The 7 Best Personal Finance Apps of 2022 (thebalancemoney.com)
Best Budgeting Apps Of October 2022 – Forbes Advisor
How to Build a Personal Finance App like Mint (relevant.software)
A Guide for Developing an Effective Personal Finance Application (appinventiv.com)

IDEATION & PROPOSED SOLUTION

6.Empathy Map Canvas



7. Ideation & Brainstorming

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

- **Team gathering** — Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- **Set the goal** — Think about the problem you'll be focusing on solving in the brainstorming session.
- **Learn how to use the facilitation tools** — Use the Facilitation Superpowers to run a happy and productive session.

Open article

Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Meganathan V

Users must be able to edit their expenses whenever required.

Automatic payments option to be provided for recurring payments.

Users must be able to classify their expenses. eg: food, entertainment, movies etc.

Resetting transactions of monthly and yearly payments.

Reward users for achieving monthly or yearly targets.

Adithya Sriram

If a group expense needs to be tracked, then add individual participants to a group similar to a social media group.

Users must be able to set monthly limits for their expenditure tracking

User can be an individual or a group of people for a common goal.

Periodic reports of monthly and yearly expenses must be generated when the user wishes to view it.

This is a title...

Lochan N

show visual charts and graphs for monthly expenses so that it is easy for users to visualize their spending habits.

User must input his/her salary per month and balance if he/she wishes to.

Aditya

predicting the expenditure for the coming months based on the previous expenditure data.

when a new expense is being added the category of the expense can be an existing one or can be added by the user.

User must be provided with the option to turn on and off the reminders for payments

Users must have an option to get their monthly expenditure analytics through mail.

For recurring payments, there must be reminders for the unpaid payments

If there are EMI's of loans to be paid every month, then application should update it automatically and should not involve the user, for the period the loan exits.

Auto tracking of payments by integrating the application with the users bank account.

3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Users must be able to edit their expenses whenever required.

Users must be able to set monthly limits for their expenditure tracking

If there are EMI's of loans to be paid every month, then application should update it automatically and should not involve the user, for the period the loan exits.

Fundament Requirements

Users must be able to classify their expenses. eg: food, entertainment, movies etc.

User can be an individual or a group of people for a common goal.

when a new expense is being added the category of the expense can be an existing one or can be added by the user.

For recurring payments, there must be reminders for the unpaid payments

Reward users for achieving monthly or yearly targets.

User must be provided with the option to turn on and off the reminders for payments

Automatic payments option to be provided for recurring payments.

Enhance UX

If a group expense needs to be tracked, then add individual participants to a group similar to a social media group.

User must be provided with the option to turn on and off the reminders for payments

predicting the expenditure for the coming months based on the previous expenditure data.

Users must have an option to get their monthly expenditure analytics through mail.

show visual charts and graphs for monthly expenses so that it is easy for users to visualize their spending habits.

Auto tracking of payments by integrating the application with the users bank account.

Additional Requirements

User must input his/her salary per month and balance if he/she wishes to.

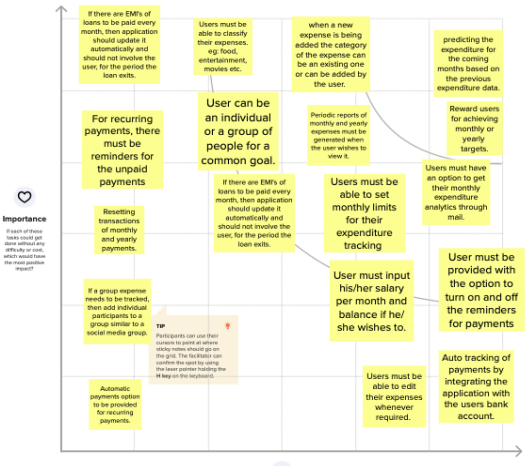
Periodic reports of monthly and yearly expenses must be generated when the user wishes to view it.

Automatic payments option to be provided for recurring payments.

4 Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- **Share the mural** — **Share a view link** to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- **Export the mural** — Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- **Strategy blueprint** — Define the components of a new idea or strategy. Open the template
- **Customer experience journey map** — Understand customer needs, motivations, and obstacles for an experience. Open the template
- **Strengths, weaknesses, opportunities & threats** — Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan. Open the template

Share template feedback

8.Proposed Solution

Date	19 September 2022
Team ID	PNT2022TMID53163
Project Name	Personal Expense Tracker
Team Members	Aditya, Adithya Sriram , Meganathan, Lochan

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Personal finance management is an important part of people's lives. But everyone do not have the knowledge or time to manage their finances in a proper manner. Even if the person has the knowledge and time for it, they do not bother tracking their expenses as they find it tedious and time-consuming. People use conventional method of maintaining records of the expenses or keep it in memory. This can lead to wrong conclusions about their spending and might end up not have money when they are in an emergency situation.
2.	Idea / Solution description	Our team aims to develop a User-friendly and a customizable Personal Expense Tracker that requires minimal effort from the user side to track their expenditure. We aim to do so through user-defined categories and goals that he/she needs to achieve. The expenditure is tracked and shown to the user Through graphs which is easy to read and interpret.

3.	Novelty / Uniqueness	• There are several application to track expenses of the user. Our project aims to make an application that is customizable , user-friendly and gives the best user experience. • Remainders for paying pending payments and also can automate the process of recurring payment. • Users can set personal financial goals and suggestions would be given by our application to achieve their goals.
4.	Social Impact / Customer Satisfaction	This application will help our users track their expenses effortlessly. Customers will be put at ease when it comes to their spendings as the only requirement is for the user to update the application when an expense occurs. The social impact of the web application would be far-reaching as the application is applicable to users of different strata's. With its ability to be customized, the application is developed to suit the needs of all our users alike.
5.	Business Model (Revenue Model)	• **Revenue-model:** Subscription fees, Unlocking premium features, expanding the storage of the application etc. • **End-users:** Students, Interested Individuals, Family Members, Working Professionals. • **Marketing:** The application will be publicized through the usage

		of various social media platforms and through word of mouth. As users begin to use the application, ratings in Google, would increase, resulting in a huge influx of customers.
6.	Scalability of the Solution	Since our application will be hosted in the IBM Cloud, scalability of the application would not be an issue. As it will be pay-as-you-use model, The expense of the application maintenance will depend on the traffic on our application and the resources used by our application to run on the cloud. And also containerizing our application will make our application decoupled and further help in scaling only required components of our application.

9.Problem solution fit:

Explain CS fit and introduce CL	**CUSTOMER SEGMENTS** CS Students/working professionals consider adopting a user-friendly application with features for budgeting.	**CUSTOMER LIMITATIONS** CL The application they want may not be secure and not all the services may be free. They want an application that is device friendly because different customer segments may use different devices like smartphones, iPads etc.	**AVAILABLE SOLUTIONS** AS Solutions are offered with the ability to link bank accounts for precise and understandable deposit statistics. Some current solutions offer advice on how to limit spending. Recurring bills are likewise tracked by existing solutions.
Concentrate on PR, use BE comprehend RC	**PROBLEMS** PR Maintaining a manual record of spending is the most efficient way to do so. The majority of people lack the drive to achieve this. Recognizing reoccurring expenses lessens repetitious labour, which pays for the user's menial chores.	**PROBLEMS/ROOT CAUSE** PR Users are reluctant to perform time-consuming, pointless calculations, which makes keeping track of spending difficult. They never get the opportunity to visualise and change their purchasing patterns as a result. The aforementioned rationale makes the natural technique of keeping track of spending less effective.	**BEHAVIOUR** BE Users anticipate that calculations will be made while they enter their spending and deposits in the background. They demand tight budgetary restrictions. They want to visualise their spending patterns and comprehend where they might make savings.
Find the robust TR and EM	**TRIGGERS TO ACT** TR Earn benefits for budgeting. Keep track of every dollar you spend. Recognize the parallel between want and need. Be assured that you will pay your payments on time.	**YOUR SOLUTION** SL Users can modify the personal spending tracker to fit their needs by making it configurable. By offering user defined spending categories, rewards, goals, and limits, we hope to achieve this. Users of the application will also have the option to view a graphical analysis of their expenditures in order to better understand their spending habits.	**CHANNELS OF BEHAVIOUR** CH **Offline** Other students are exposed to it when they use and discuss their expertise in class. A team of experts is involved for businesses, and through word-of-mouth other businesses and people will learn about this application.
	EMOTIONS EM Before: The user lacks confidence in their ability to manage their spending and is aimless in doing so. Missing bills is more common for the user. After: The user has more control over their spending. He or she is less likely to forget to pay a bill since they feel rewarded for cutting costs.		**Online** The use of various social media channels will be used to sell the application. As consumers start using the software, the App Store's ratings would rise, bringing in a massive flood of new users.

REQUIREMENT ANALYSIS

10. Functional Requirements:

A **Functional Requirement** is a description of the service that the software must offer. It describes a software system or its component.

FR No.	Functional Requirement	Functional Requirement Description
FR 1	User Login	User needs to login through app's user interface and user authentication is done on the server side connected to IBM DB2.
FR 2	User Registration	A new user needs to register through the app and it is connected to the IBM DB2.
FR 3	Tracking income and expenses	Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards)
FR 4	Transaction Receipts:	Capture and organize your payment receipts to keep track of your expenditure.
FR 5	Payments & Invoices:	Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the tracking app sends reminders for payments and automatically matches the payments with invoices.

FR 6	Detailed Reports:	The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.
FR 7	In-depth insights and analytics:	Provides in-built tools to generate reports with easy-to-understand visuals and graphics to gain insights about the performance of your business.
FR 8	Recurrent Expenses:	Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.
FR 9	Budget Vs. Actual Spent:	The user gets a detailed insight into the real-time income and expenditure. Thus, you can plan your budget strategically to reduce unnecessary expenses.
FR 10	Prediction:	With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. It brings out the factors causing fluctuations in your expenses.

11. Non Functional Requirements:

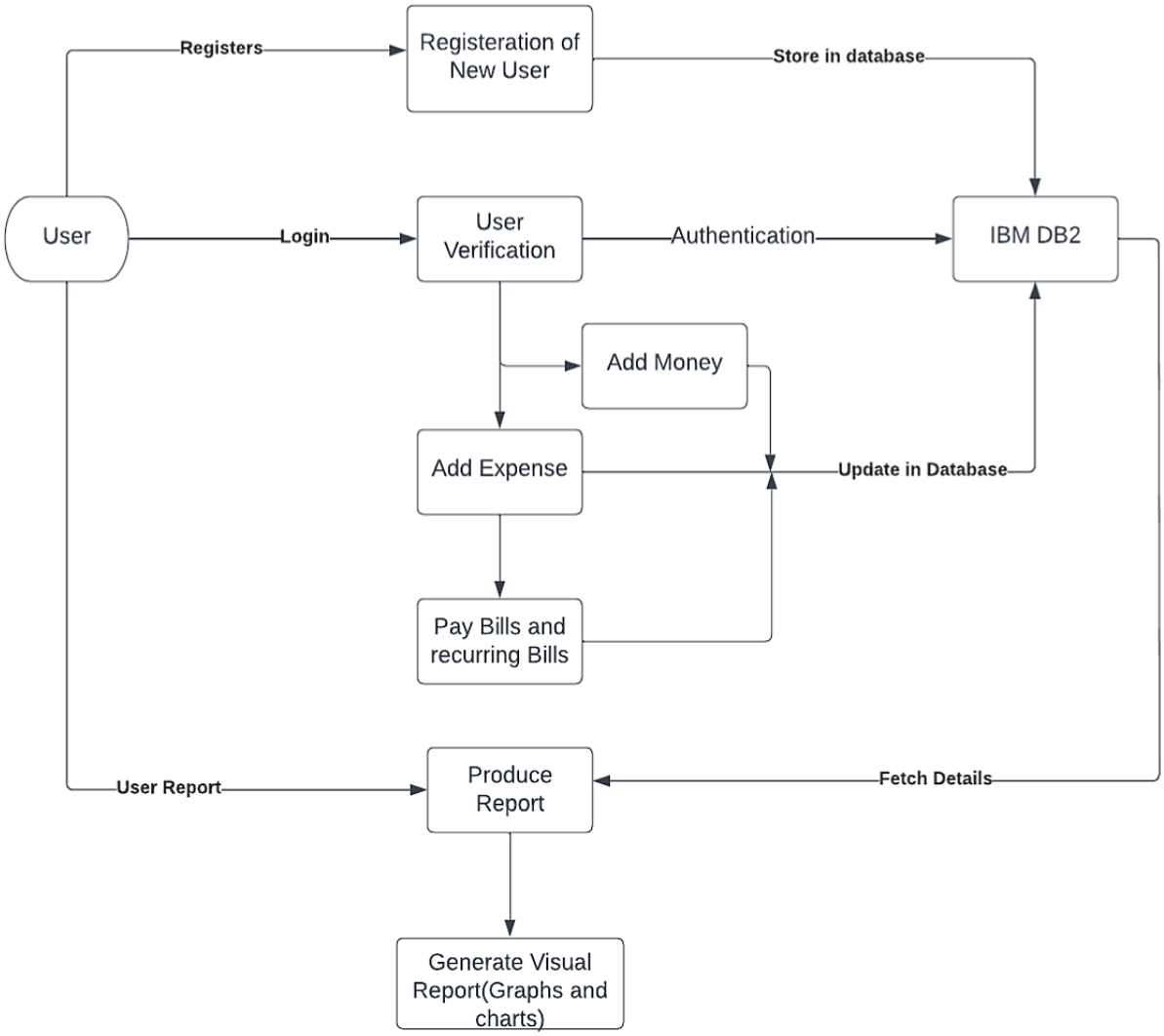
Non-Functional Requirement(NFR) specifies the quality attribute of a

software system.

NRF No.	Non-Functional Requirement	Description
NRF 1	Security	User Information such as login information and data of expenditure must be stored securely in the cloud. Access to IBM DB 2 must be restricted and data leak must be avoided.
NRF 2	Reliability	Our application is reliable which addresses fault tolerance and proof to software failure.
NRF 3	Performance	Our application will provide upmost performance as it is a cloud application and also its response time is minimum.
NRF 4	Availability	Our application is available to the user 99.9% of the year as it is hosted on IBM DB2 which provides good back up to failures.
NRF 5	Scalability	Our application is hosted on IBM DB2 which will be a cloud application and it is easy to scale on cloud based on the demand of the application.

PROJECT DESIGN

12. Data Flow Diagrams


Registers
Registeration of New User
Store in database
User
Login
User Verification
Authentication
IBM DB2
Add Money
Add Expense
Update in Database
Pay Bills and recurring Bills
User Report
Produce Report
Fetch Details
Generate Visual Report(Graphs and charts)

13.Solution & Technical Architecture

Project Design Phase-II

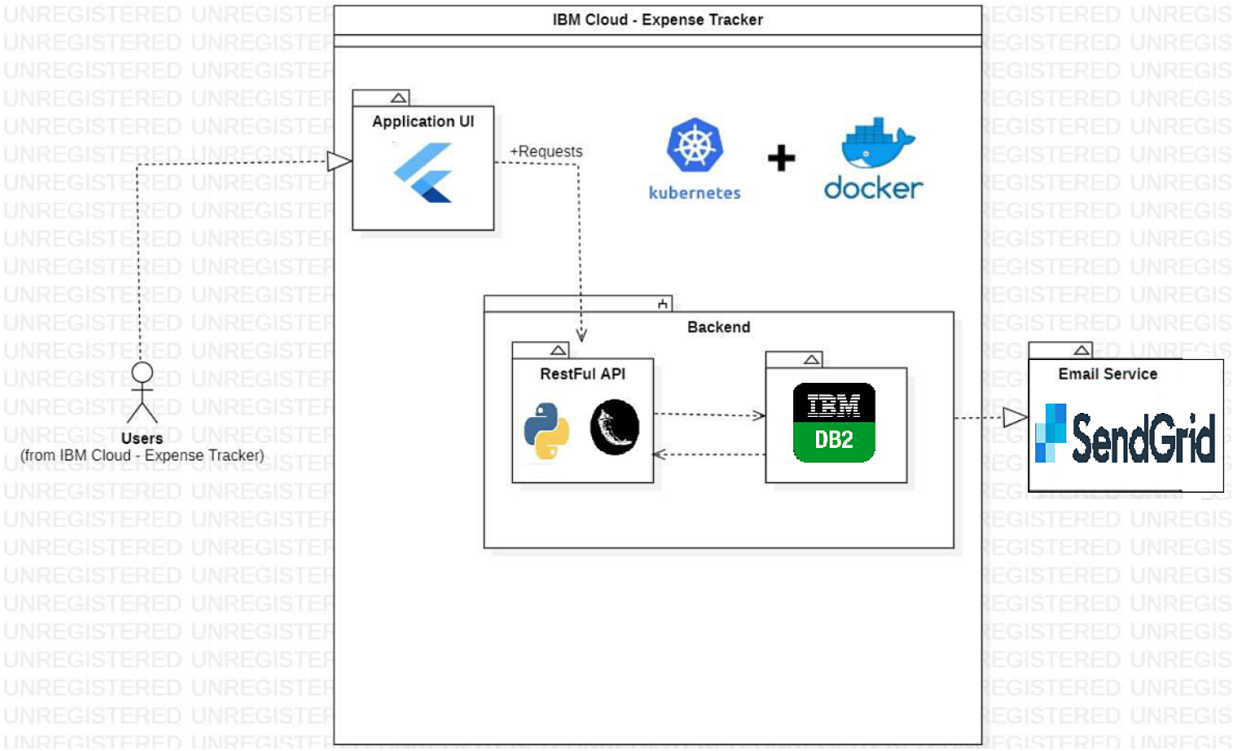


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User interface for accessing the features of the web app	Flutter
2.	Application Logic -Mail	Sending mail to a user in	Sendgrid API

	Service	case of any important event	
3.	Application Logic -Add and View Expenses	Users must be able to add new expenses while being able to delete ,update or view old ones.	Python: Flask
4.	Application Logic - Spending cap	Whenever the budget limit for a cycle is crossed, user must be notified.	Python: Flask
5.	Database	Data Type, Configurations etc.	IBM DB2
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Source Code of tools available to public	All tools used for development are open source by nature.
2.	Security Implementations	Maintaining Data confidentiality and Authorization	Bcrypt for hashing and AES for encryption , HTTPS for overall security during transmission.
3.	Scalable Architecture	Use of microservices ensures scalability in business logic.	Flask- microservices architecture
4.	Availability	Cloud Application serves very well in Scalability and Availability.	Kubernetes for maintaining scalability in deployment.

5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

14. User Stories

User Stories

Functional Requirement (Epic)	**User Story Number**	**User Story /Task**	**Priority**
Registration Page	IM-1	As a user I should be able to register to the app	High
Login Page	IM-2	As a user I should be able to login to the app	High
Profile Page	IM-3	As a user I should be able to access my profile page	High
Profile Page	IM-4	As a user I should be able to edit my details	Medium
Profile Page	IM-5	As a user I should be able to logout	High
Home Page	IM-6	As a user I should be able to add my expenses	Medium
Home Page	IM-7	As a user I should be able to see my daily expenditure	High
Home Page	IM-8	As a user I should be able to edit or delete my expenses	High

Income Page	IM-9	As a user I should be able to add my income	Medium
Income Page	IM-10	As a user I should be able to edit my income	Medium
Expenses Page	IM-11	As a user I should be able to set my cycle size	High
Expenses Page	IM-12	As a user I should be able to set my threshold limit	High
Expenses Page	IM-13	As a user I should be notified if for over spent	High
Deployment	IM-14	As a user I should be able to monitor different categories of expense	Medium
Deployment	IM-15	As a user I should be able to use the app on the cloud platform	High
Deployment+Home Page	IM-16	As a user I should be able to see the graphical representation of my expenses	High

PROJECT PLANNING & SCHEDULING

15. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Members
Sprint-1	Registration Page	IM-1	As a user I should be able to register to the app	2	High	Adithya Sriram R, Aditya R
Sprint-1	Login Page	IM-2	As a user I should be able to login to the app	2	High	Aditya R, Meganathan V
Sprint-1	Profile Page	IM-3	As a user I should be able to access my profile page	2	High	Lochan N, Meganathan V
Sprint-1	Profile Page	IM-4	As a user I should be able to edit my details	2	Medium	Adithya Sriram R, Lochan N
Sprint-1	Profile Page	IM-5	As a user I should be able to logout	2	High	Meganathan V, Aditya R
Sprint-2	Home Page	IM-6	As a user I should be able to add my expenses	3	Medium	Aditya R, Adithya Sriram R
Sprint-2	Home Page	IM-7	As a user I should be able to see my daily expenditure	2	High	Lochan N, Meganathan V

Sprint-2	Home Page	IM-8	As a user I should be able to edit or delete my expenses	2	High	Lochan N, Aditya R
Sprint-2	Income Page	IM-9	As a user I should be able to add my income	2	Medium	Adithya Sriram R, Meganathan V
Sprint-2	Income Page	IM-10	As a user I should be able to edit my income	2	Medium	Aditya R, Lochan N
Sprint-3	Expenses Page	IM-11	As a user I should be able to set my cycle size	3	High	Adithya Sriram R, Aditya R
Sprint-3	Expenses Page	IM-12	As a user I should be able to set my threshold limit	3	High	Meganathan V, Adithya Sriram R
Sprint-3	Expenses Page	IM-13	As a user I should be notified if for over spent	3	High	Lochan N, Meganathan V
Sprint-4	Deployment	IM-14	As a user I should be able to monitor different categories of expense	2	Medium	Meganathan, Aditya R
Sprint-4	Deployment	IM-15	As a user I should be able to use the app on the cloud platform	3	High	Aditya R, Lochan N

Sprint-4	Deployment+Home Page	IM-16	As a user I should be able to see the graphical representation of my expenses	3	High	Adithya Sriram R, Lochan N

16. Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	1. (Meet Planned Date)
Sprint-2	11	6 Days	30 Oct 2022	4 Nov 2022	11	2. (Meet Planned Date)

Sprint-3	9	6 Days	6 Nov 2022	12 Nov 2022	9	3. (Meet Planned Date)
Sprint-4	10	6 Days	12 Nov 2022	18 Nov 2022	8	4. (Meet Planned Date)

Velocity:

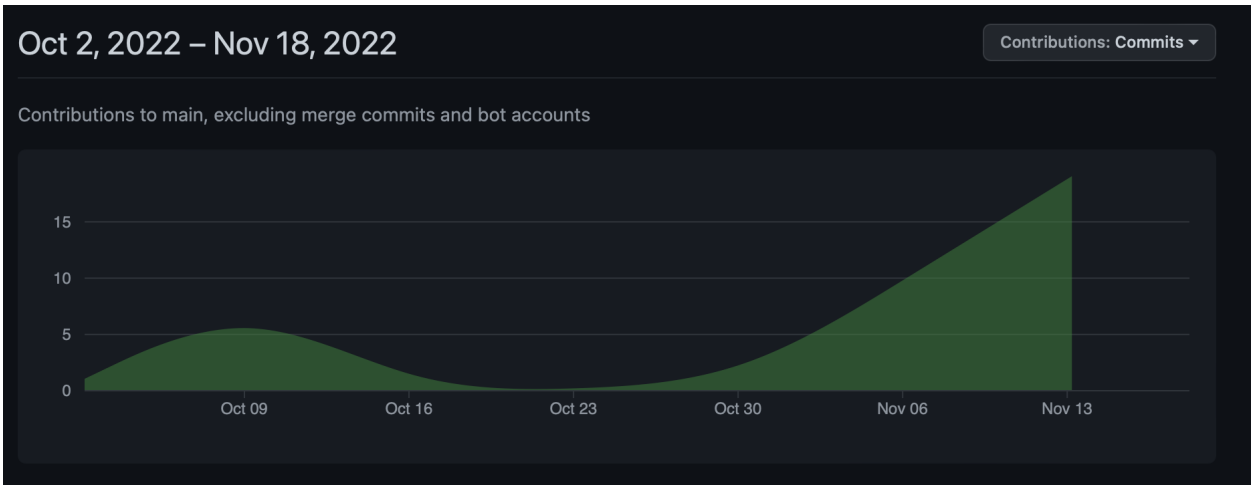
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

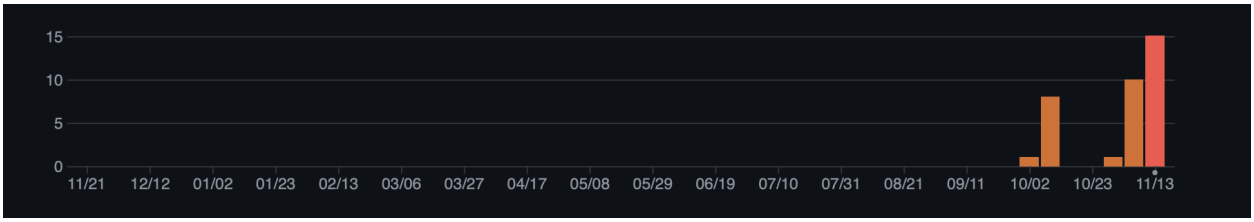
Average Velocity:

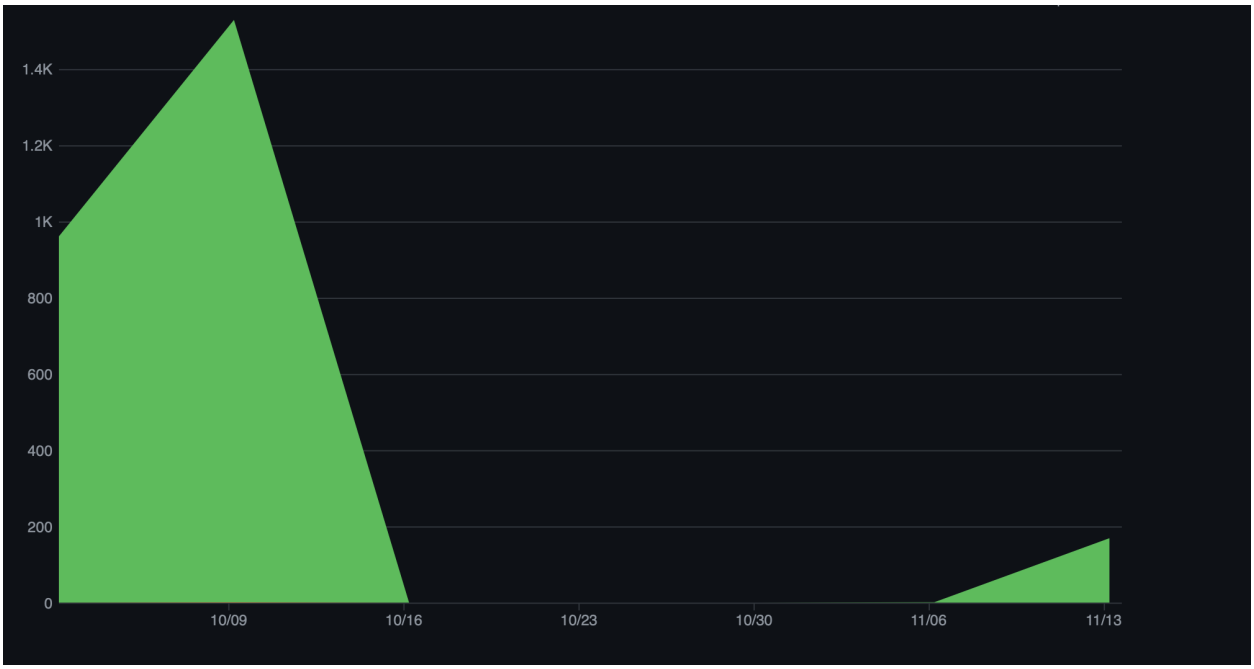
Average Points per sprint = (10+11+9+8)/4=9.5

Story points per day/average velocity = 9.5/6.5=1.461

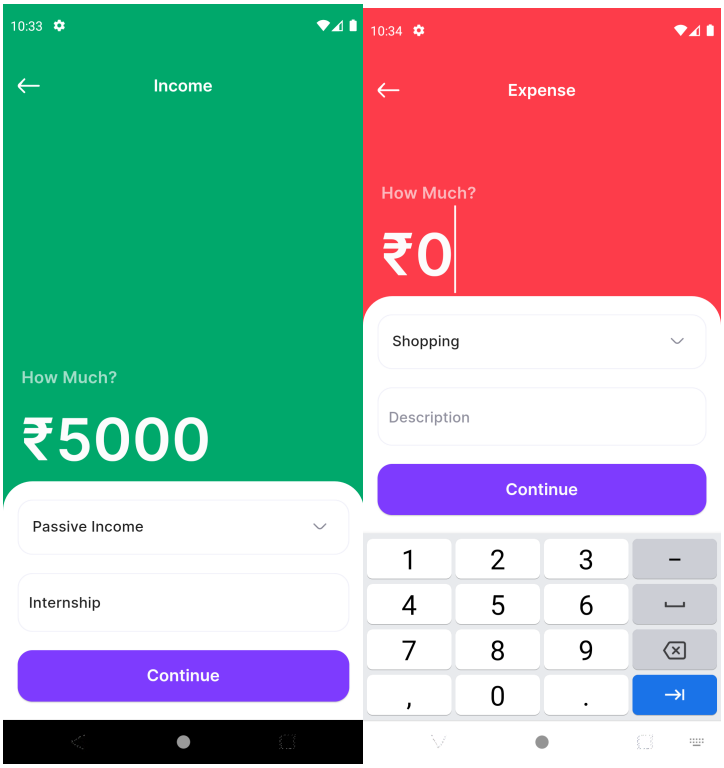
17. Analytics from JIRA







18 Add, Remove Transaction




```dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
```

```dart
import 'package:flutter_svg/flutter_svg.dart';
import 'package:personal_expense_tracker/constants.dart';
import
'package:personal_expense_tracker/controllers/transaction_controller.
dart';
import 'package:personal_expense_tracker/models/transaction.dart';

class AddExpenseScreen extends ConsumerStatefulWidget {
  const AddExpenseScreen({Key? key}) : super(key: key);

  @override
  ConsumerState createState() => _AddExpenseScreenState();
}

class _AddExpenseScreenState extends ConsumerState<AddExpenseScreen>
{
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController amountTextEditingController =
      TextEditingController(text: "0");
  Categories selectedCategory = Categories.shopping;
  String description = "";

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        FocusScope.of(context).unfocus();
      },
      child: Scaffold(
        backgroundColor: red100,
        appBar: AppBar(
          leading: IconButton(
            onPressed: () {
              Navigator.pop(context);
            },
            icon: SvgPicture.asset(
              "assets/icons/arrow_left.svg",
              color: Colors.white,
            ),
```

```
          ),
          title: const Text(
            "Expense",
            style: TextStyle(color: Colors.white),
          ),
        ),
        body: Form(
          key: _formKey,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.end,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Padding(
                padding: const EdgeInsets.only(left: 20),
                child: Text(
                  "How Much?",
                  style: title3TextStyle.copyWith(
                    color: light80.withOpacity(0.64),
                  ),
                ),
              ),
              Padding(
                padding: const EdgeInsets.only(left: 20),
                child: TextFormField(
                  controller: amountTextEditingController,
                  style: moneyInputTextStyle,
                  decoration: moneyTextInputDecoration,
                  cursorColor: Colors.white,
                  keyboardType: TextInputType.number,
                  textInputAction: TextInputAction.next,
                  onChanged: (val) {
                    if (val.startsWith("0")) {
                      amountTextEditingController.text =
                          "${int.tryParse(val) ?? "0"}";
                      amountTextEditingController.selection =
                          TextSelection.collapsed(
                              offset:
amountTextEditingController.text.length);
                    }
```

```
                  },
                ),
              ),
              DecoratedBox(
                decoration: const BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.vertical(
                    top: Radius.circular(32),
                  ),
                ),
                child: Padding(
                  padding:
                      const EdgeInsets.symmetric(horizontal: 16,
vertical: 20),
                  child: Column(
                    children: [
                      ButtonTheme(
                        alignedDropdown: true,
                        child: DropdownButtonFormField<Categories>(
                          value: selectedCategory,
                          decoration: textInputDecoration.copyWith(
                            contentPadding:
                                const EdgeInsets.fromLTRB(0, 20, 12,
16),
                          ),
                          isExpanded: true,
                          icon:
SvgPicture.asset("assets/icons/dropdown.svg"),
                          items: const [
                            DropdownMenuItem(
                              value: Categories.shopping,
                              child: Text("Shopping"),
                            ),
                            DropdownMenuItem(
                              value: Categories.subscription,
                              child: Text("Subscription"),
                            ),
                            DropdownMenuItem(
                              value: Categories.food,
```

```
                              child: Text("Food"),
                            ),
                          ],
                          onChanged: (val) {
                            selectedCategory = val ??
selectedCategory;
                          },
                        ),
                      ),
                      const SizedBox(height: 20),
                      TextFormField(
                        cursorColor: light20,
                        decoration: textInputDecoration.copyWith(
                          hintText: "Description",
                        ),
                        validator: (val) => (val?.isEmpty ?? false)
                            ? "Please enter a description"
                            : null,
                        onChanged: (val) {
                          description = val;
                        },
                      ),
                      const SizedBox(height: 20),
                      ElevatedButton(
                        onPressed: () {
                          if (_formKey.currentState!.validate()) {
                            ref

.read(transactionListStateProvider.notifier)
                                .addTransaction(
                                    double.parse(

amountTextEditingController.text),
                                    selectedCategory,
                                    description);
                            Navigator.of(context).pop();
                          }
                        },
                        child: const Text("Continue"),
```

```
                      ),
                    ],
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

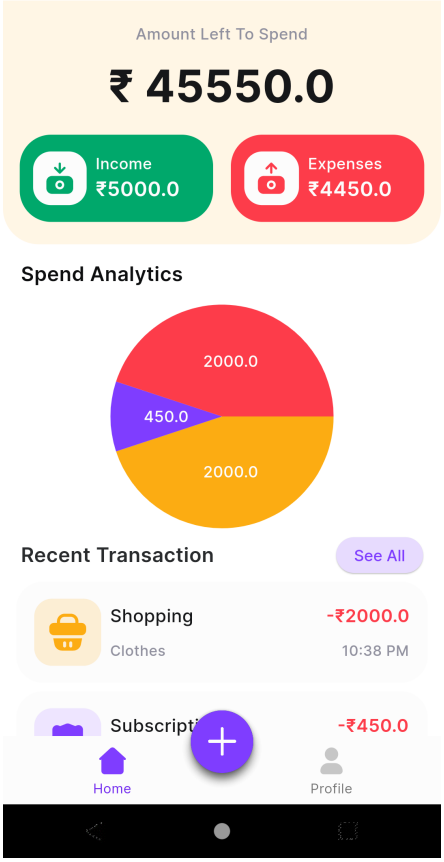
```
Future addTransaction(
    double amount, Categories category, String description) async {
  final response = await http.post(
    Uri.parse("http://10.0.2.2:5000/transaction/add/"),
    body: {
      'amount': amount.toString(),
      'category': category.toString(),
      'description': description,
      'email': ref.read(authStateProvider)?.email,
    },
  );
  if (response.statusCode == 200) {
    state = [...state,
Transaction.fromJson(jsonDecode(response.body))];
  }
}

Future deleteTransaction(int tId) async {
  final response = await http.get(
    Uri.parse("http://10.0.2.2:5000/transaction/delete/$tId"),
  );
  if (response.statusCode == 200) {
    state = state.where((transaction) => transaction.tId !=
tId).toList();
```

```
  }
}
```

```
@app.route("/transaction/add/",methods=['POST'])
def addtransaction():
    now = datetime.now()
    category = request.form['category']
    description = request.form['description']
    amount = request.form['amount']
    email = request.form['email']
    insert_sql = "INSERT INTO Transaction
(CATEGORY,DESCRIPTION,AMOUNT,TRANSACTIONDATE,EMAIL) VALUES
(?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, category)
    ibm_db.bind_param(prep_stmt, 2, description)
    ibm_db.bind_param(prep_stmt, 3, amount)
    ibm_db.bind_param(prep_stmt, 4, now)
    ibm_db.bind_param(prep_stmt, 5, email)
    ibm_db.execute(prep_stmt)

    sql = "SELECT * FROM Transaction WHERE TransactionDate=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, now)
    ibm_db.execute(stmt)
    insertedTransaction = ibm_db.fetch_assoc(stmt)


transaction={"tId":insertedTransaction["TID"],"amount":float(amount),"
category":category,"description":description,"transactionDate":str(no
w)}
    return jsonify(transaction),200

@app.route("/transaction/delete/<tId>")
def deletetransaction(tId):
    sql = "DELETE FROM Transaction WHERE Tid =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, tId)
```

```
    ibm_db.execute(stmt)
    return "Deleted Successfully",200
```

19 Graphical Representation



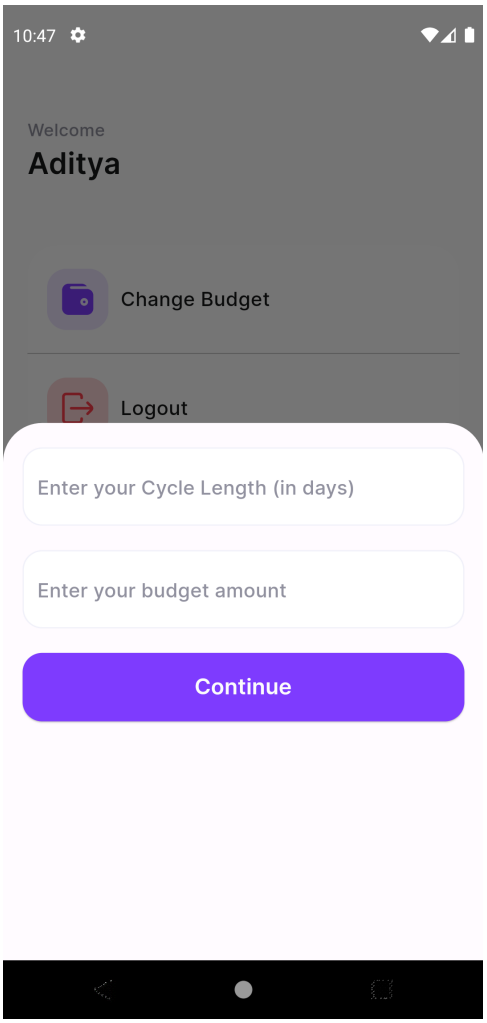

```
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:personal_expense_tracker/constants.dart';
import
'package:personal_expense_tracker/controllers/transaction_controller.
dart';
import 'package:personal_expense_tracker/models/transaction.dart';
```

```dart
class ExpenseChart extends ConsumerStatefulWidget {
  const ExpenseChart({Key? key}) : super(key: key);

  @override
  ConsumerState createState() => _ExpenseChartState();
}

class _ExpenseChartState extends ConsumerState<ExpenseChart> {
  int touchedIndex = -1;
  @override
  Widget build(BuildContext context) {
    final transactionList = ref.watch(transactionListStateProvider);
    double shoppingAmount = 0;
    double foodAmount = 0;
    double transportAmount = 0;
    double subscriptionAmount = 0;

    for (int i = 0; i < transactionList.length; i++) {
      switch (transactionList[i].category) {
        case Categories.shopping:
          shoppingAmount += transactionList[i].amount;
          break;
        case Categories.subscription:
          subscriptionAmount += transactionList[i].amount;
          break;
        case Categories.food:
          foodAmount += transactionList[i].amount;
          break;
        case Categories.transport:
          transportAmount += transactionList[i].amount;
          break;
      }
    }

    return SizedBox(
      height: 200,
      child: PieChart(
        PieChartData(
          centerSpaceRadius: 0,
```

```
        sectionsSpace: 0,
        sections: [
          PieChartSectionData(
            radius: (touchedIndex == 0) ? 110 : 100,
            color: yellow100,
            value: shoppingAmount,
            titleStyle: body3Light80TextStyle,
            showTitle: !(touchedIndex == 0),
            badgeWidget: (touchedIndex == 0)
                ? const Text(
                    "Shopping",
                    style: body3Light80TextStyle,
                  )
                : null,
          ),
          PieChartSectionData(
            radius: (touchedIndex == 1) ? 110 : 100,
            color: violet100,
            value: subscriptionAmount,
            titleStyle: body3Light80TextStyle,
            showTitle: !(touchedIndex == 1),
            badgeWidget: (touchedIndex == 1)
                ? const Text(
                    "Subscription",
                    style: body3Light80TextStyle,
                  )
                : null,
          ),
          PieChartSectionData(
            radius: (touchedIndex == 2) ? 110 : 100,
            color: red100,
            value: foodAmount,
            titleStyle: body3Light80TextStyle,
            showTitle: !(touchedIndex == 2),
            badgeWidget: (touchedIndex == 2)
                ? const Text(
                    "Food",
                    style: body3Light80TextStyle,
                  )
```

```
                  : null,
            ),
            PieChartSectionData(
              radius: (touchedIndex == 3) ? 110 : 100,
              color: blue100,
              value: transportAmount,
              titleStyle: body3Light80TextStyle,
              showTitle: !(touchedIndex == 3),
              badgeWidget: (touchedIndex == 3)
                  ? const Text(
                      "Transport",
                      style: body3Light80TextStyle,
                    )
                  : null,
            )
          ],
          pieTouchData: PieTouchData(
            touchCallback: (FlTouchEvent event, pieTouchResponse) {
              setState(() {
                if (!event.isInterestedForInteractions ||
                    pieTouchResponse == null ||
                    pieTouchResponse.touchedSection == null) {
                  touchedIndex = -1;
                  return;
                }
                touchedIndex =

pieTouchResponse.touchedSection!.touchedSectionIndex;
              });
            },
          ),
        ),
        swapAnimationDuration: const Duration(milliseconds: 250),
        swapAnimationCurve: Curves.easeIn,
      ),
    );
  }
}
```

20. Update Cycle Length and Budget Amount



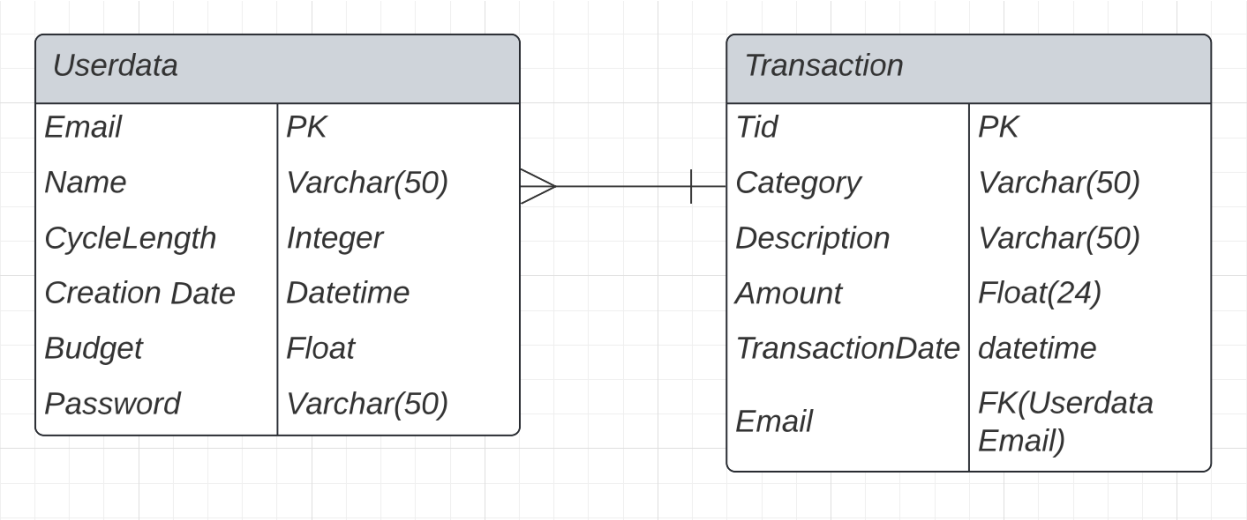

```dart
int cycleLength = 30;
double budget = 30000;
await showModalBottomSheet(
  context: context,
  isDismissible: false,
  shape: const RoundedRectangleBorder(
    borderRadius: BorderRadius.vertical(
      top: Radius.circular(32),
    ),
  ),
```

```
  builder: (context) => Padding(
    padding: const EdgeInsets.symmetric(
      horizontal: 16,
      vertical: 20,
    ),
    child: Column(
      children: [
        TextFormField(
          cursorColor: light20,
          keyboardType: TextInputType.number,
          decoration: textInputDecoration.copyWith(
            hintText: "Enter your Cycle Length (in days)",
          ),
          onChanged: (val) {
            cycleLength = int.tryParse(val) ?? cycleLength;
          },
        ),
        const SizedBox(height: 20),
        TextFormField(
          cursorColor: light20,
          keyboardType: TextInputType.number,
          decoration: textInputDecoration.copyWith(
            hintText: "Enter your budget amount",
          ),
          onChanged: (val) {
            budget = double.tryParse(val) ?? budget;
          },
        ),
        const SizedBox(height: 20),
        ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text("Continue"),
        ),
      ],
    ),
  ),
);
```

```
await ref
    .read(authStateProvider.notifier)
    .updateBudget(cycleLength, budget);
```

```
@app.route('/updateBudget/', methods=['POST'])
def updateBudget():
    insert_sql = "UPDATE Userdata SET CycleLength =?, Budget =? WHERE
Email =?"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1,request.form['cycleLength'])
    ibm_db.bind_param(prep_stmt, 2,request.form['budget'])
    ibm_db.bind_param(prep_stmt, 3,request.form['email'])
    ibm_db.execute(prep_stmt)
    return "Budget Updated Successfully",200
```

21. Database Schema



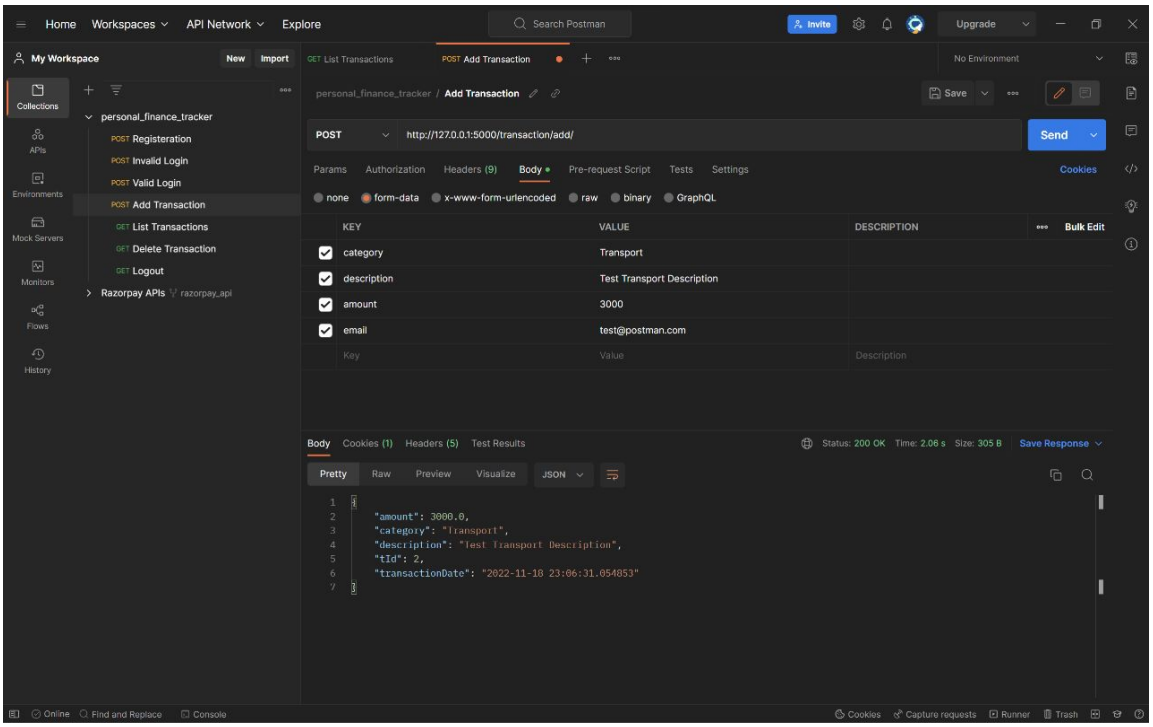
Testing

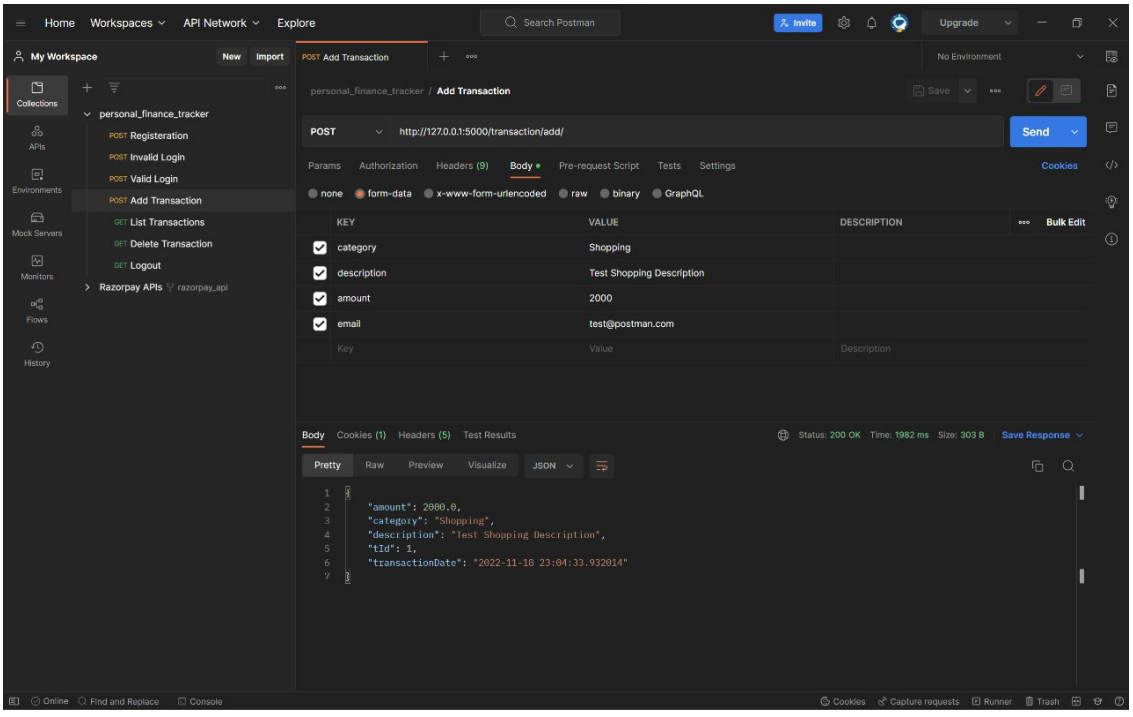
22. Test Cases

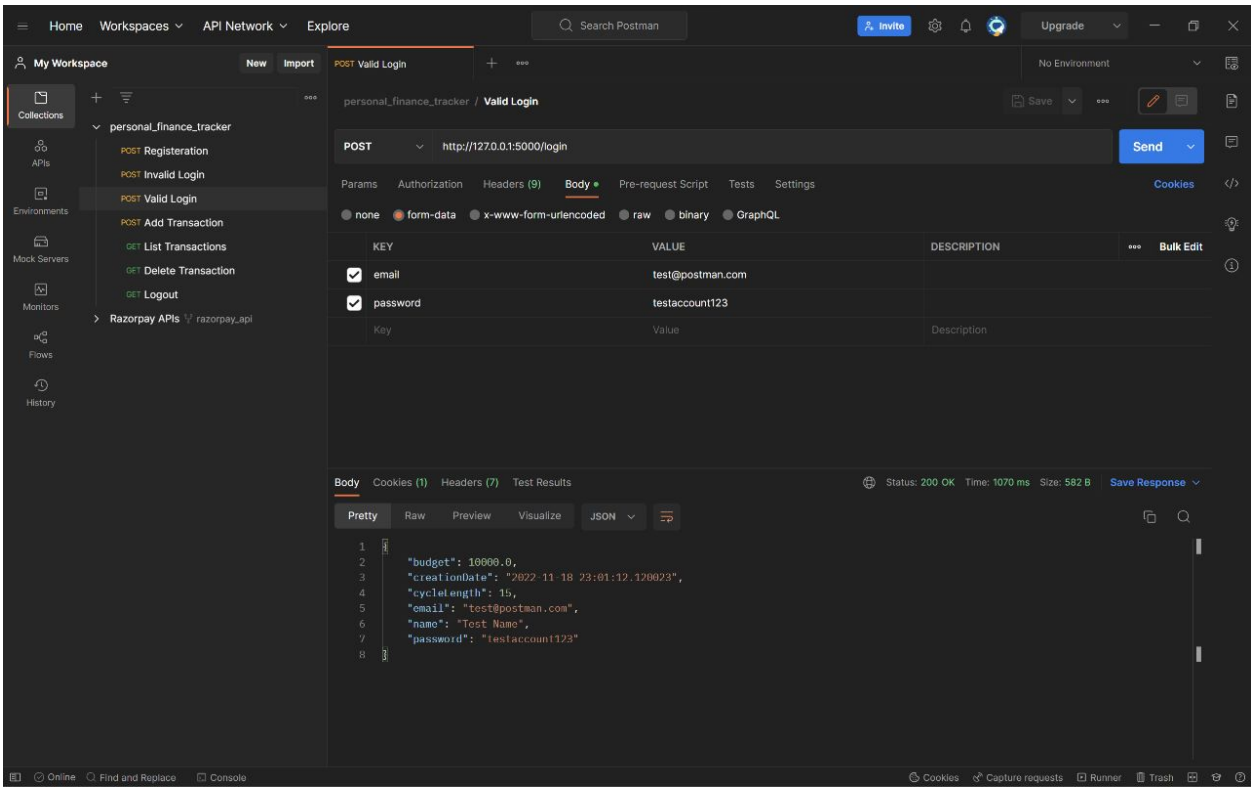
- Verify that the user is able to login successfully on entering appropriate credentials.
- The UI elements, such as the card for the login, the button to submit etc are functional and are rendered properly in all devices.
- Users who enter invalid credentials will not be redirected to the dashboard
- Verify that users are able to register themselves to the application.
- The UI elements, such as the card for the registration, the button to submit etc arefunctional and are rendered properly in all devices.
- The user should not be able to register successfully if any of the fields are left empty.
- The UI part of the dashboard, the side navbar, the logout option, the cards showing various expenses, and the income edit icon must be functioning and rendered properly.
- The UI part of the add expense page must be rendered and functioning.
- The user must be able to add an expense.
- The user should be able to update the balance and it must be reflected in the dashboard where the wallet balance is displayed.
- The user views all the UI components rendered properly and functioning accordingly.
- The user should be able to set a monthly limit for his expenditures.
- The page must render properly and function appropriately.
- The user must be able to view the analysis of their expenses.
- The user must be able to view the analysis of their expenses.
- The UI is rendered properly and the various components are functioning properly.
- This page allows the user to view the recurring expenses that he or she has

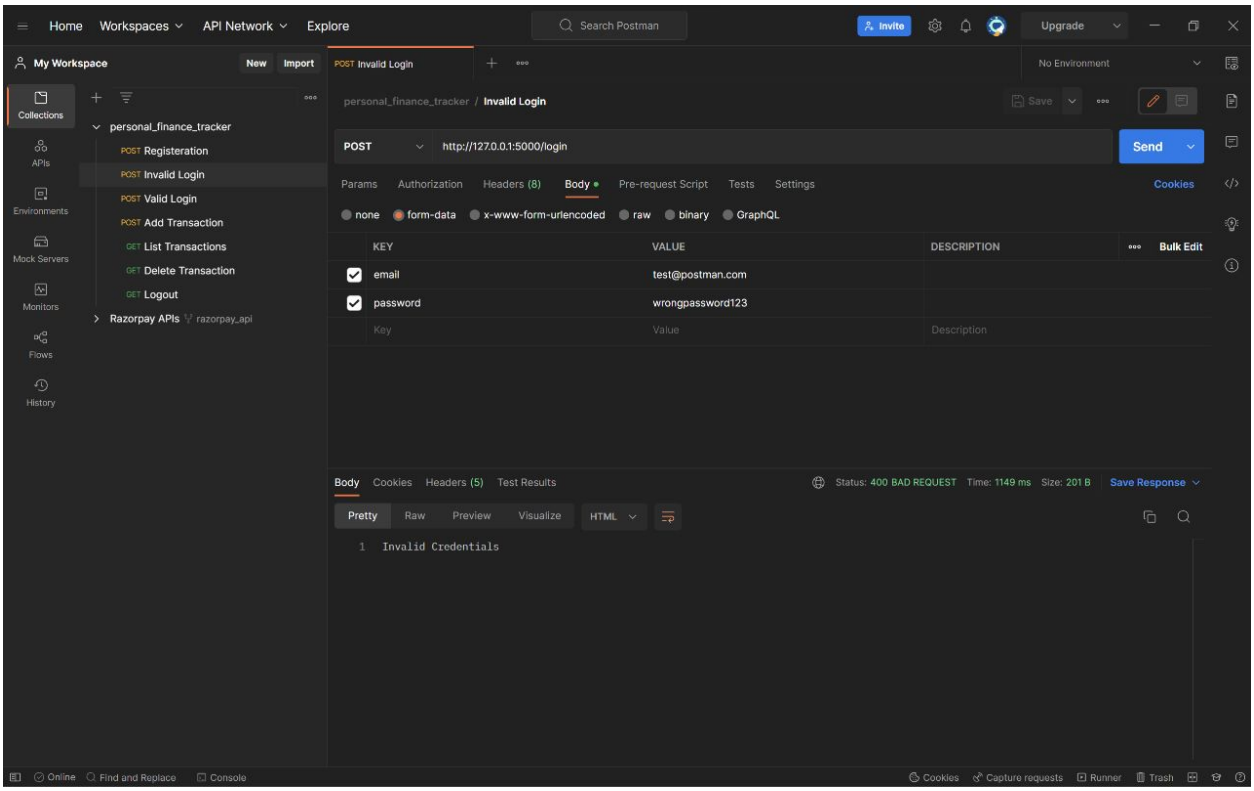
created using the add recurring expense page.

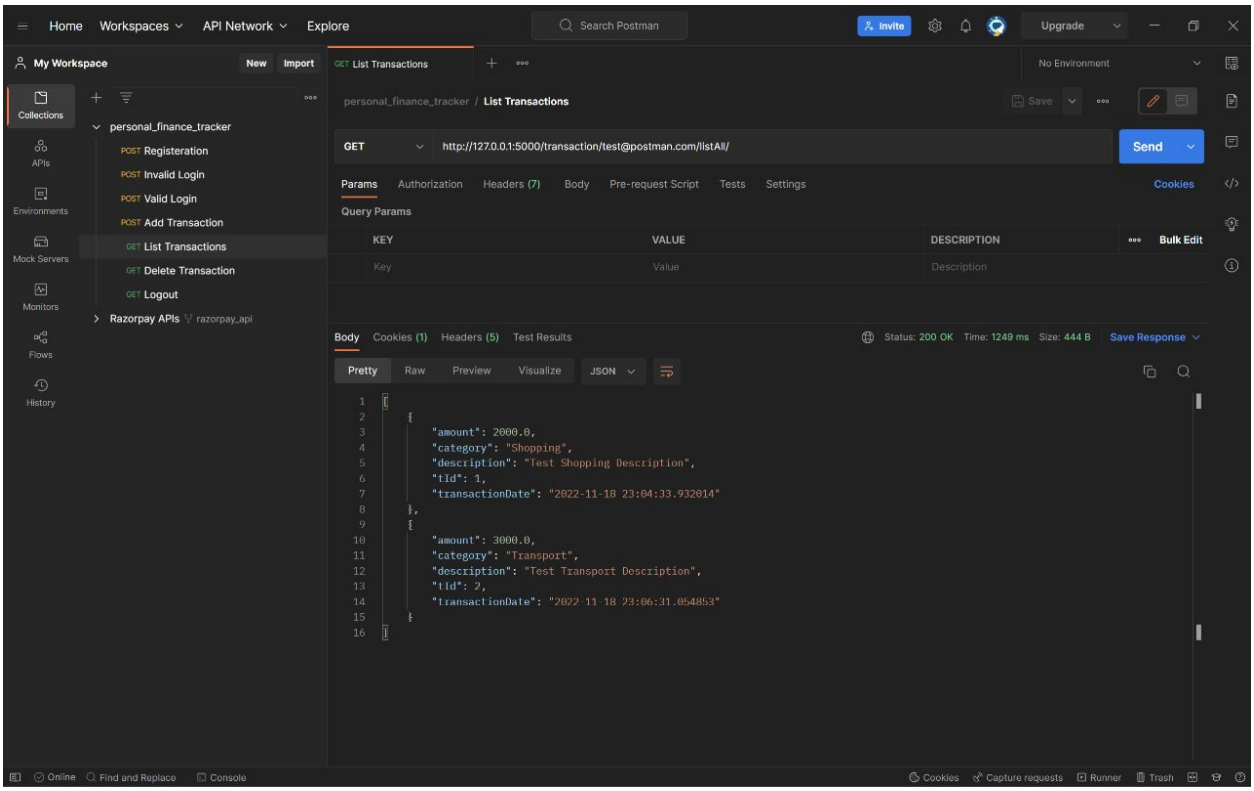
Adding Transactions(Testing using Postman)

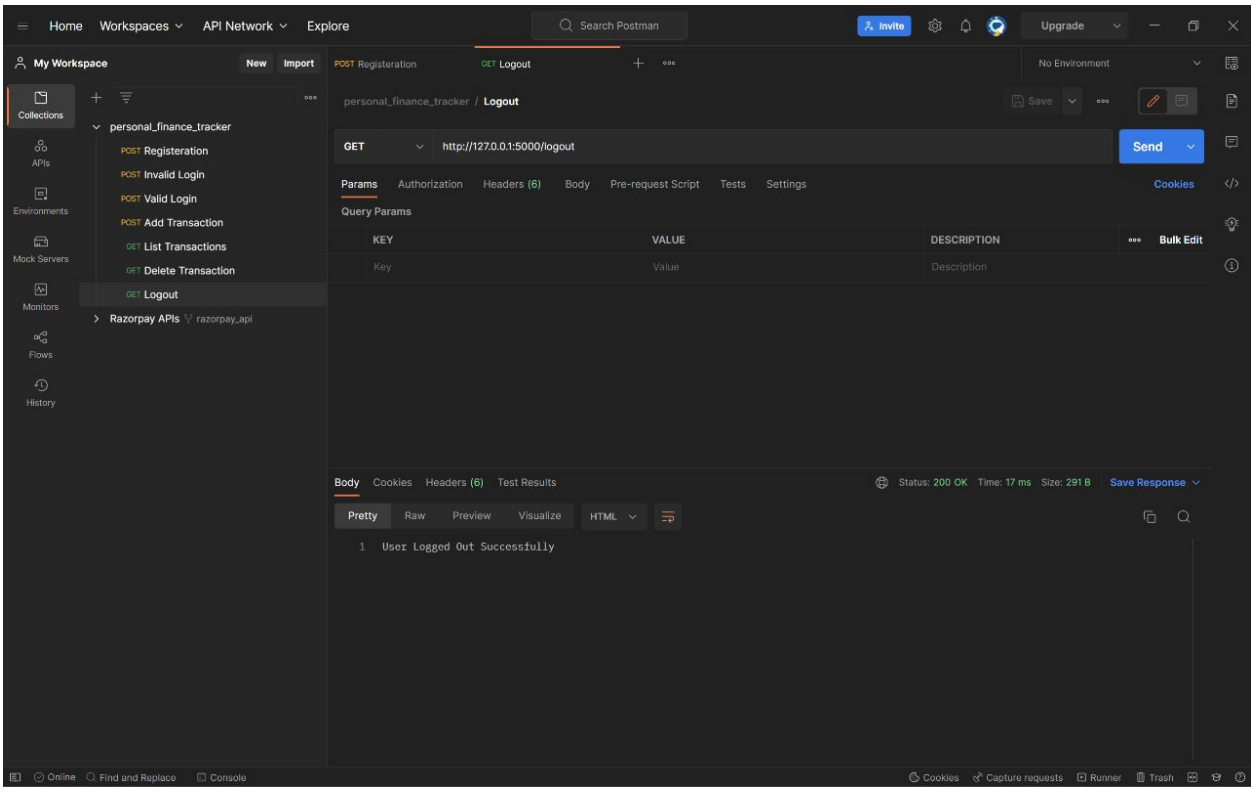


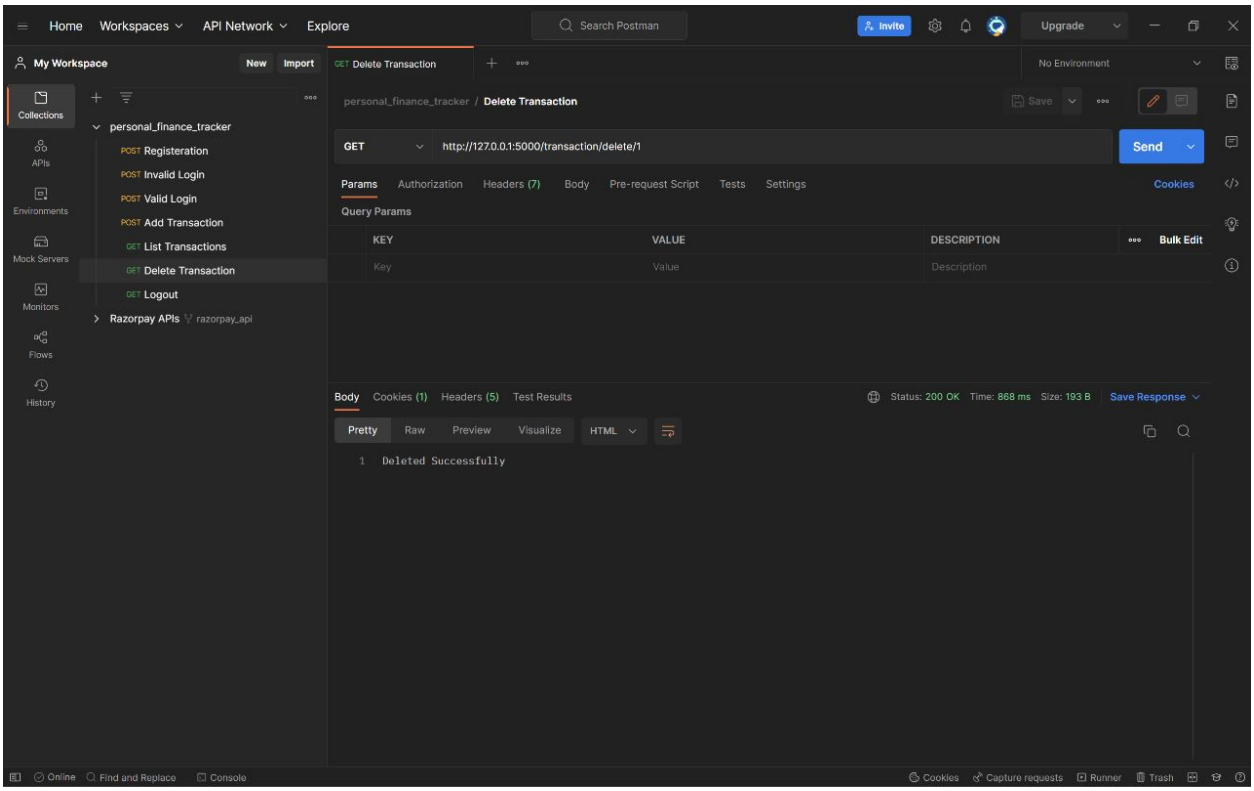


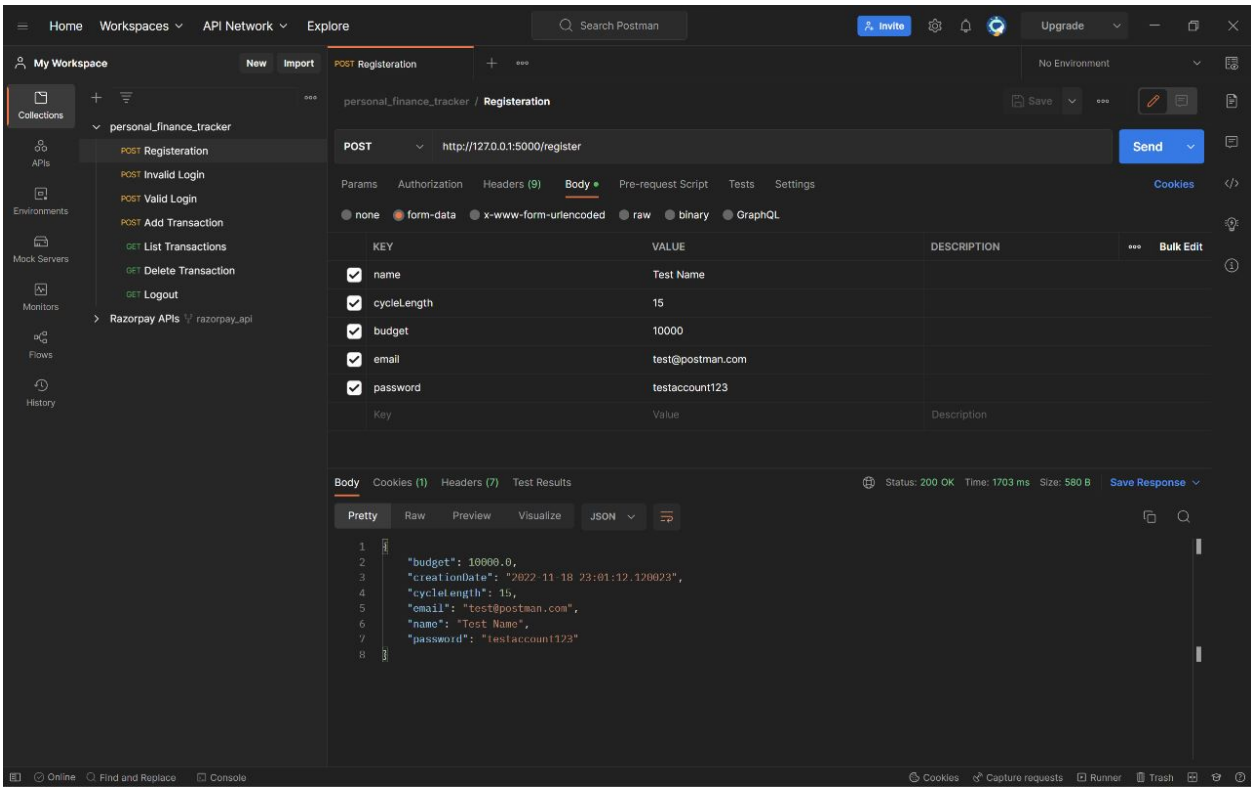

Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Valid Login
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / Valid Login
Save
POST
http://127.0.0.1:5000/login
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
email
test@postman.com
password
testaccount123
Key
Value
Description
Body
Cookies (1)
Headers (7)
Test Results
Status: 200 OK
Time: 1070 ms
Size: 582 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"budget": 10000.0,
"creationDate": "2022-11-18 23:01:12.120023",
"cycleLength": 15,
"email": "test@postman.com",
"name": "Test Name",
"password": "testaccount123"
}
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Invalid Login
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / Invalid Login
Save
POST
http://127.0.0.1:5000/login
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
email
test@postman.com
password
wrongpassword123
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 400 BAD REQUEST
Time: 1149 ms
Size: 201 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
Invalid Credentials
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
GET List Transactions
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / List Transactions
Save
GET
http://127.0.0.1:5000/transaction/test@postman.com/listAll/
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies (1)
Headers (5)
Test Results
Status: 200 OK
Time: 1249 ms
Size: 444 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"amount": 2000.0,
"category": "Shopping",
"description": "Test Shopping Description",
"tId": 1,
"transactionDate": "2022-11-18 23:04:33.932014"
},
{
"amount": 3000.0,
"category": "Transport",
"description": "Test Transport Description",
"tId": 2,
"transactionDate": "2022-11-18 23:06:31.054853"
}
]
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Registeration
GET Logout
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / Logout
Save
GET
http://127.0.0.1:5000/logout
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 17 ms
Size: 291 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
User Logged Out Successfully
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
GET Delete Transaction
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / Delete Transaction
Save
GET
http://127.0.0.1:5000/transaction/delete/1
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies (1)
Headers (5)
Test Results
Status: 200 OK
Time: 868 ms
Size: 193 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Deleted Successfully
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Registeration
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
personal_finance_tracker
POST Registeration
POST Invalid Login
POST Valid Login
POST Add Transaction
GET List Transactions
GET Delete Transaction
GET Logout
Razorpay APIs
razorpay_api
personal_finance_tracker / Registeration
Save
POST
http://127.0.0.1:5000/register
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
name
Test Name
cycleLength
15
budget
10000
email
test@postman.com
password
testaccount123
Key
Value
Description
Body
Cookies (1)
Headers (7)
Test Results
Status: 200 OK
Time: 1703 ms
Size: 580 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"budget": 10000.0,
"creationDate": "2022-11-18 23:01:12.120023",
"cycleLength": 15,
"email": "test@postman.com",
"name": "Test Name",
"password": "testaccount123"
}
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash

23.User Acceptance Testing

UAT Execution & Report Submission

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issuesof the Personal Expense Tracker project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report showsthe number of resolved or closed bugs at each severity level, and how they were resolved

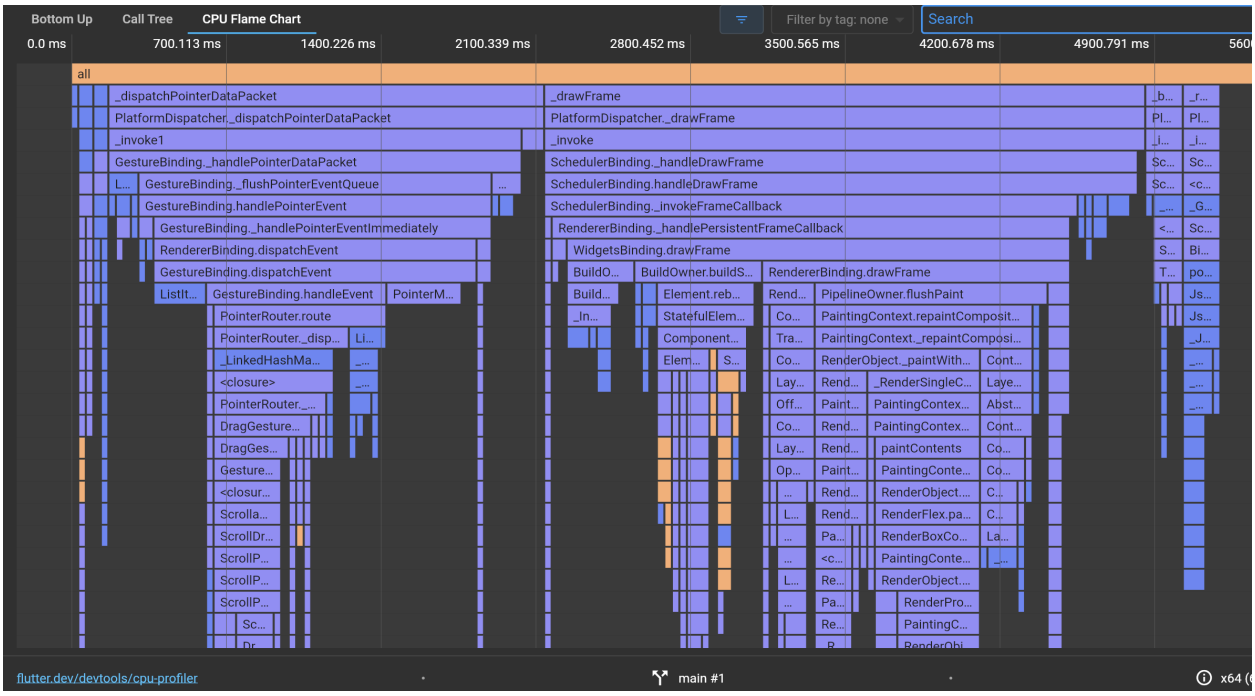
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	0	0	0
External	1	0	0	0	1
Fixed	0	2	0	0	2
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	1	2	0	0	3

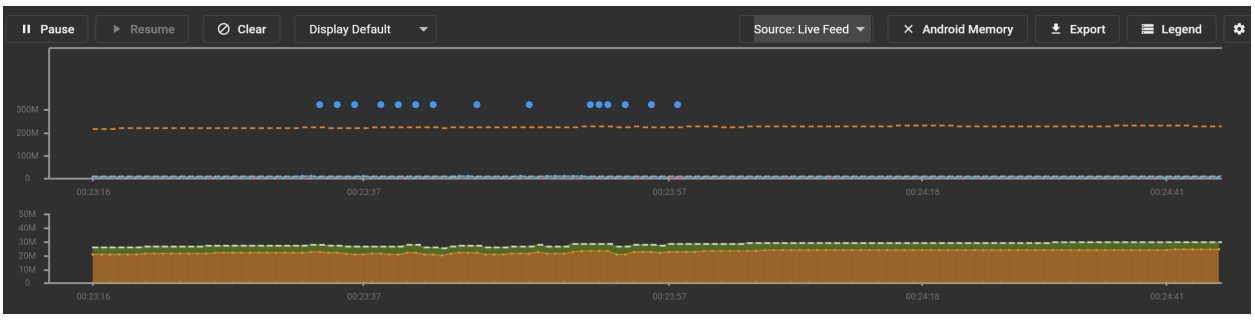
3. Test Case Analysis

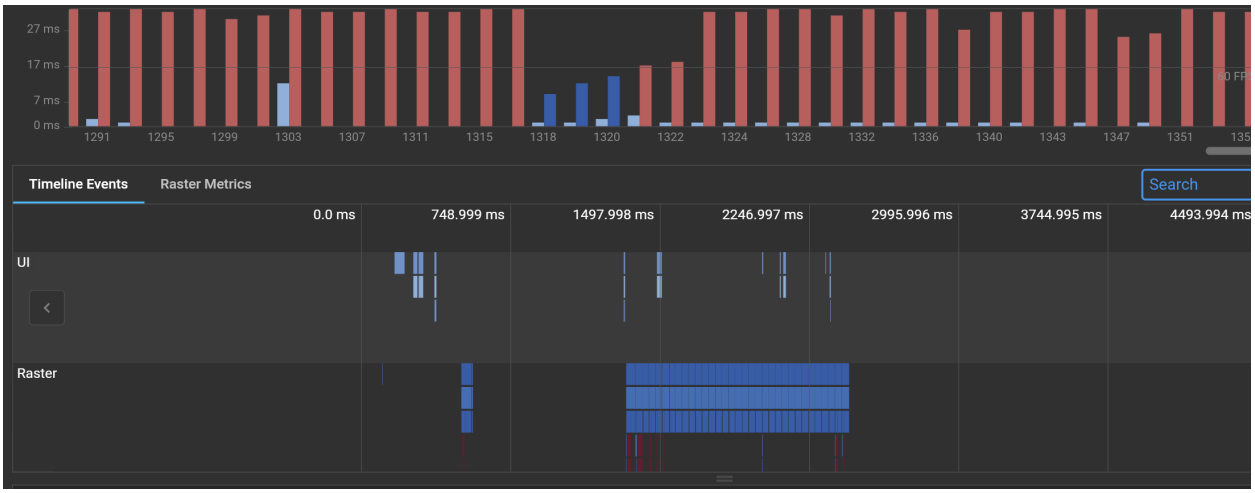
This report shows the number of test cases that have passed, failed,and untested

Section	Total Cases	Not Tested	Fail	Pass
Login	3	0	0	3
Registration	4	0	0	4
Dashboard	1	0	0	1
Expense Addition	3	0	1	2
Update Balance	2	0	0	2
Set threshold limit	2	0	0	2
View Analysis	3	0	1	2
Set cycle size	2	0	0	2
Category Creation	2	0	0	2
Modifying an Expense	2	0	0	2

23. Performance Metrics


Bottom Up
Call Tree
CPU Flame Chart
Filter by tag: none
Search
0.0 ms
700.113 ms
1400.226 ms
2100.339 ms
2800.452 ms
3500.565 ms
4200.678 ms
4900.791 ms
all
_dispatchPointerDataPacket
_drawFrame
PlatformDispatcher._dispatchPointerDataPacket
PlatformDispatcher._drawFrame
_invoke1
_invoke
GestureBinding._handlePointerDataPacket
SchedulerBinding._handleDrawFrame
GestureBinding._flushPointerEventQueue
SchedulerBinding.handleDrawFrame
GestureBinding.handlePointerEvent
SchedulerBinding._invokeFrameCallback
GestureBinding._handlePointerEventImmediately
RendererBinding._handlePersistentFrameCallback
RendererBinding.dispatchEvent
WidgetsBinding.drawFrame
GestureBinding.dispatchEvent
BuildOwner.buildS...
RendererBinding.drawFrame
GestureBinding.handleEvent
PointerM...
Element.reb...
PipelineOwner.flushPaint
PointerRouter.route
StatefulElem...
PaintingContext.repaintComposit...
PointerRouter._disp...
Component...
PaintingContext._repaintComposi...
_LinkedHashMa...
RenderObject._paintWith...
<closure>
PointerRouter._...
_RenderSingleC...
DragGesture...
PaintingContex...
DragGes...
paintContents
Gesture...
PaintingConte...
<closur...
RenderObject....
Scrolla...
RenderFlex.pa...
ScrollDr...
RenderBoxCo...
ScrollP...
RenderObject....
RenderPro...
Sc...
PaintingC...
flutter.dev/devtools/cpu-profiler
main #1
x64


Pause
Resume
Clear
Display Default
Source: Live Feed
Android Memory
Export
Legend
300M
200M
100M
0
00:23:16
00:23:37
00:23:57
00:24:18
00:24:41
50M
40M
30M
20M
10M


27 ms
17 ms
7 ms
0 ms
1291
1295
1299
1303
1307
1311
1315
1318
1320
1322
1324
1328
1332
1336
1340
1343
1347
1351
Timeline Events
Raster Metrics
Search
0.0 ms
748.999 ms
1497.998 ms
2246.997 ms
2995.996 ms
3744.995 ms
4493.994 ms
UI
Raster

ADVANTAGES & DISADVANTAGES

Advantages:

1) Writing down every expense encourages more careful spending and helps you avoid splurging. It teaches you to be frugal with your money

2) You regain full control of your finances when you keep track of your expenses. You will always be aware of how much money is available for spending and how much is currently in your bank account.

3) You'll have a better understanding of what is happening with your money as you monitor your spending over time.

4) Many of your everyday costs might not seem like much, but when you total up your purchases of lottery tickets, coffee, restaurants, and other indulgences, you could be surprised at how much your habits truly cost.

5) You may create precise budgets for your monthly expenditures by keeping track of your costs. You must keep track of your spending every day once you create a budget, which is a monthly spending plan that takes your income and costs into consideration.

6) Daily cost tracking enables you to monitor your progress toward your financial objectives. Daily cost tracking enables you to monitor your progress toward your financial objectives.

7) Financial issues result from disorganised finances. Maintaining organisation is simpler than trying to clean up a chaotic financial condition.

8) It aids in keeping track of your bank accounts. What if someone stole your debit card details and began using it to make purchases with your money? You can prevent these hazards if you keep track of your spending.

Disadvantages:

1) Managers and approvers may be neglectful at times due to the simplicity of rapidly approving expense reports with the press of a mouse. Although the software automatically evaluates the reports for policy violations, the approvers may choose to disregard any violations that the software missed.
2) Although automation makes auditors' jobs easier, there is less room for auditors to look into

shady activity. Since the software currently internally verifies all cost reports, external verification may become unnecessary or negligible. As a result, auditors can decide to put their trust in the software's judgement and ignore uncertainty.
3) When you have problems, need to set up anything, or even if you simply require information, poor customer assistance can hinder your operations. Therefore, it is essential to take into account a software's reputation for customer service while selecting it.
4) Checking client reviews is a crucial consideration when selecting an expense management programme. After all, the programme must be well-liked by both your staff and the financial departments.

CONCLUSION:

Tracking your spending on a daily basis might not only help you save money, but it can also help you set financial objectives for the long run. Knowing exactly where your money goes each month will make it simple for you to identify areas where concessions and savings can be made. Compared to other income and expense trackers, our project is more effective. The manual computation, which is often done in the absence of a cost tracker, is successfully avoided by the project. The modules are created in a productive and appealing way. Sticky notes, spreadsheets, and ledgers that create confusion and data consistency issues while documenting and separating expenses will be replaced by the programme. With the help of our software, users may better control their spending and do so in a more efficient manner. Not only can keeping track of daily spending help you save money, it can also help you set future financial objectives. Knowing where our money is going each day makes it simple to make sacrifices and other decisions to help save costs. Compared to existing trackers, this project is designed to operate more quickly and eliminate tedious calculations. It is designed to be effective and aesthetically pleasing at the

FUTURE SCOPE:

1) It will offer different record-keeping choices (for example Food, Travelling Fuel, Salary etc.).

2) It will continue to give notifications for our cycle spending automatically.

3) In today's hectic and expensive world, we are in a hurry to generate money, yet at the end of

the month, we have broken off. As we naively waste money on unnecessary items and titles.

So we devised a strategy to maximise our profits.

4) The user can construct their own categories for spending types such as food, clothing, rent, and bills where they must input the money spent and can also include additional data in supplementary data to identify the expense.

APPENDIX

Github Link:- https://github.com/IBM-EPBL/IBM-Project-19645-1659703081

VideoLink: https://drive.google.com/file/d/17I5SlciwaW9yYZzy1L4LTjIOpCbMrvxL/view?usp=sharing